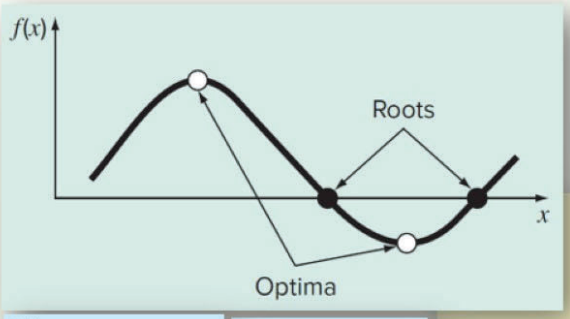
OPTIMIZATION

Textbook (Python) Part II Chapter 7

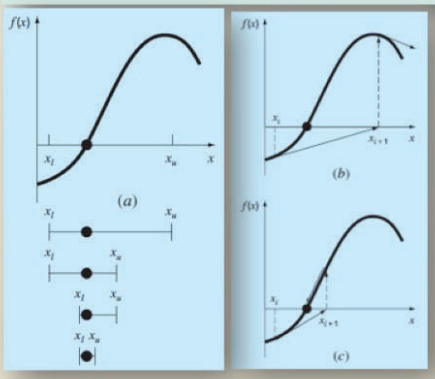
Common Q & A

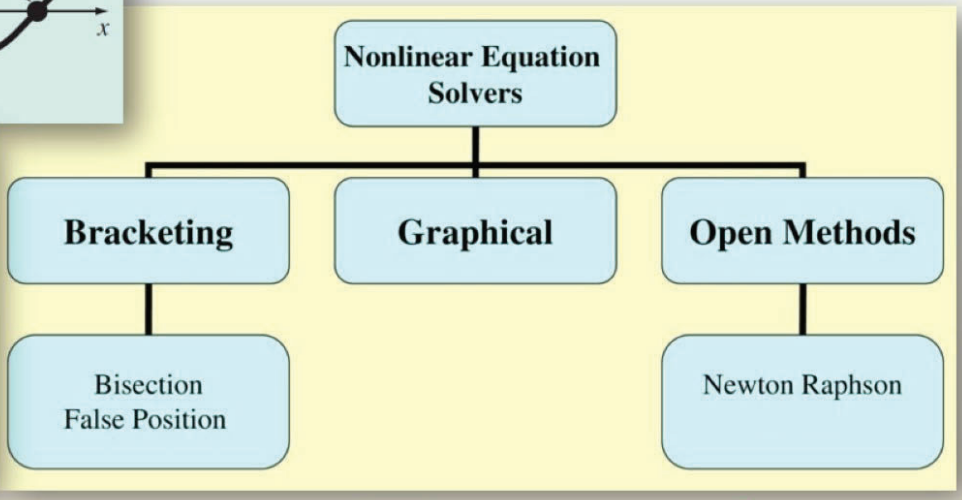
- **Errors and stopping criteria**
  - Follow the definitions given in lecture 2 slides #13 and #15
  - ε_t, ε_a, and ε_s are percentage values (though sometimes they are written without a % sign). Additionally, ε_t and ε_a can be either negative or positive.
  - Regardless of how a program is coded, always follow the definitions of errors (as noted above) when answering a question. This means that you need to be able to extract the right values from the given (textbook) source code.
- **Counting iterations**
  - The first approximation value/root calculated is iteration 1
  - Initial approximation/root (if any) is iteration 0

Solving root finding problems



- All are iterative numerical methods - use initial values to generate a sequence of improving approximate solutions





https://en.wikipedia.org/; https://slideplayer.com/slide/13272557/

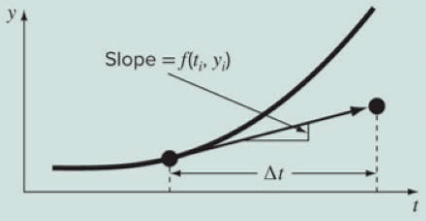
Topics

(Ordinary) differential equations

Given $\frac{dy}{dt} \approx \frac{\Delta y}{\Delta t} = f(t, y)$

solve for y as a function of t

$y_{i+1} = y_i + f(t_i, y_i)\,\Delta t$

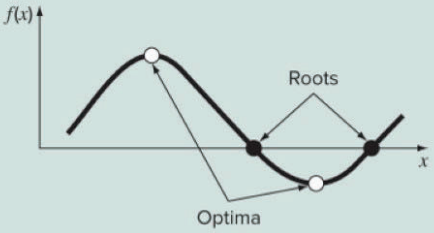


Roots and optimization

Roots: Solve for *x* so that *f*(*x*) = 0

Optimization:

Solve for *x* so that *f*′(*x*) = 0

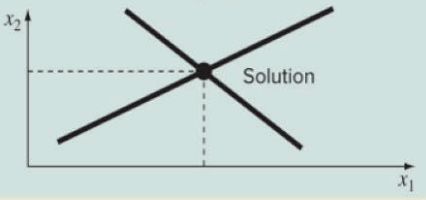


Linear algebraic equations

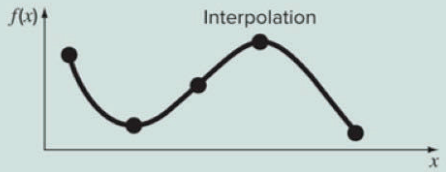
Given *a*'s and *b*'s, solve for *x*'s

$a_{11}x_1 + a_{12}x_2 = b_1$

$a_{21}x_1 + a_{22}x_2 = b_2$



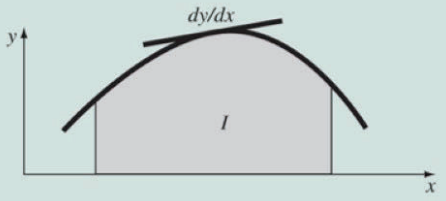
Curve fitting



Numerical integration & differentiation

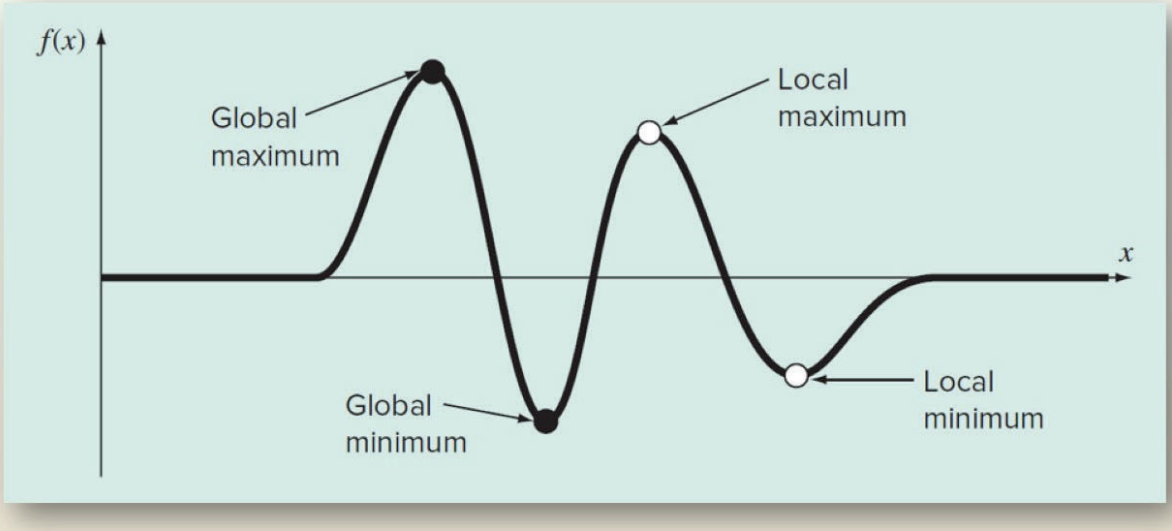
Integration: Find the area under the curve

Differentiation: Find the slope of the curve



Global vs. local optimum

- A **global optimum** represents the very best solution. A **local optimum**, though not the very best, is the best solution in a neighborhood around it.



Golden Section Search

- An **iterative method** that looks to reduce the search region by
  1. Use **golden ratio** to divide the search range into two overlapped sections
  2. Determine which one of the two sections should contains the min/max point, keep it and eliminate the other
  3. One of the end points of the region (in the middle of the new potential area) will be selected as the approximated minimum or maximum.
  4. Repeat Steps 1-3 until convergence

Golden Section Search (minimum)

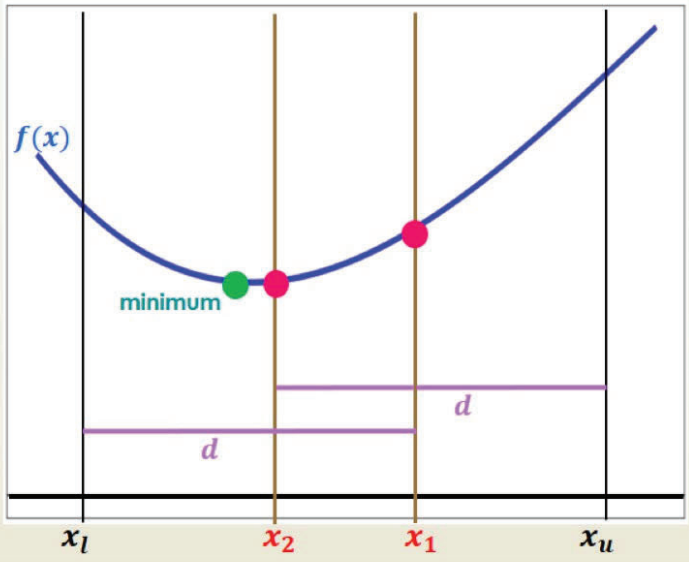
- Finding min inside $[x_l, x_u]$
- Find x_1 and x_2 where

$$x_1 = x_l + d$$

$$x_2 = x_u - d$$

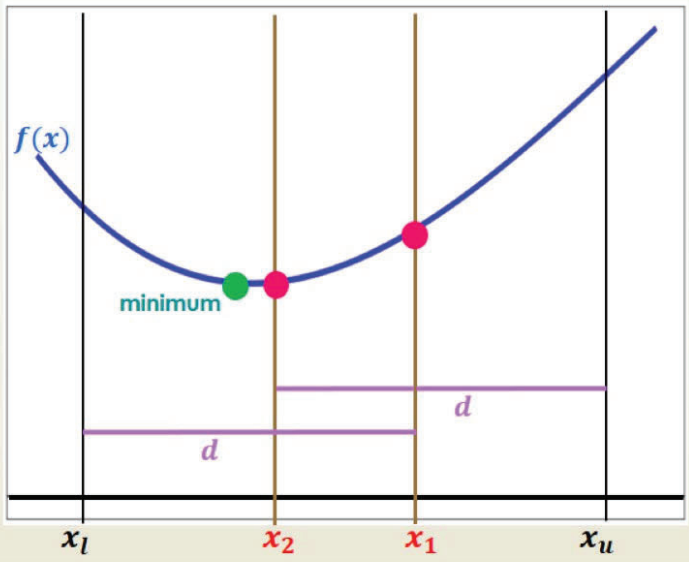
$$d = (\phi - 1)(x_u - x_l)$$

$$\phi = \frac{1+\sqrt{5}}{2} = 1.6180339$$



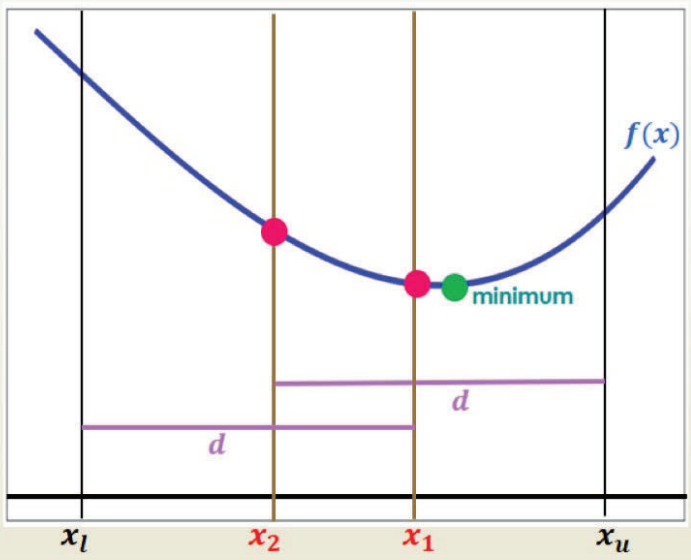
Golden Section Search (minimum)

- If $f(x_2) < f(x_1)$, then
  - the minimum must be in $[x_l, x_1]$
  - set x_2 as the new (estimated) min



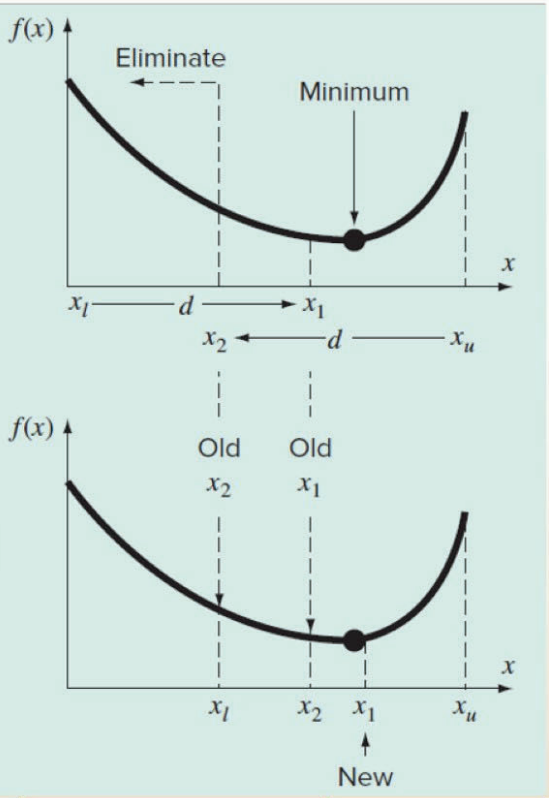
Golden Section Search (minimum)

- If $f(x_1) < f(x_2)$, then
  - the minimum must be in $[x_2, x_u]$
  - set x_1 as the new (estimated) min



Why golden ratio?

- **Save computation!**
- Because x_1 and x_2 are chosen using the golden ratio, we do not have to recalculate both function values for the next iterations.



WORKED EXAMPLES

- Use the golden section search method to find the minimum of $f(x) = x^2 + 2x$ within the interval from $x_l = -3$ to $x_u = 5$.

WORKED EXAMPLES

- Use the golden section search method to find the minimum of $f(x) = x^2 + 2x$ within the interval from $x_l = -3$ to $x_u = 5$.

iter	x_l	x_2	$f(x_2)$	x_1	$f(x_1)$	x_u	d
1							
2							
3							
4							
5							
6							

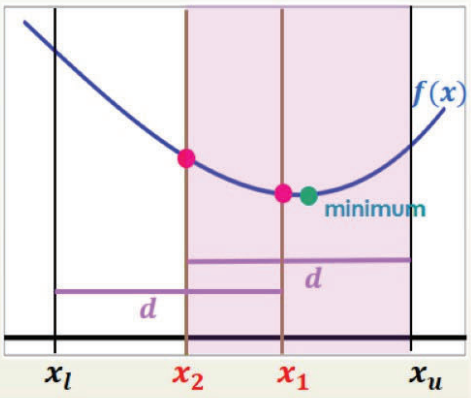
Calculating approximate errors

- Finding **the upper bound on the error**
- If the true value were at the far left, the max distance from the estimate is

$$x_1 - x_2 = (2\phi - 3)(x_u - x_l) = 0.2361(x_u - x_l)$$

- If it were at the far right, max distance is

$$x_u - x_1 = (2 - \phi)(x_u - x_l) = 0.3820(x_u - x_l)$$



- Taking the max of the two cases and normalize to the current estimate
- $\varepsilon_a = (2 - \phi)\left|\frac{x_u - x_l}{x_{opt}}\right| \times 100$

Note: since the interior points are symmetrical, either case can be used to defined errors

Golden Section Search (maximum)

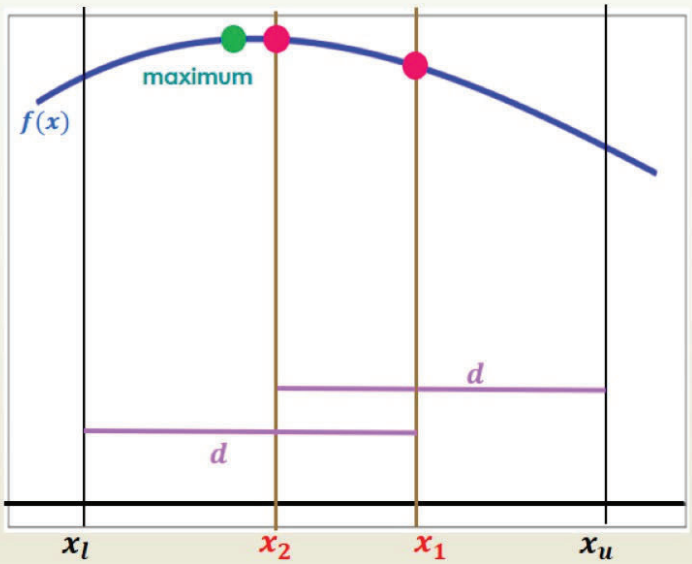
- Finding max inside $[x_l, x_u]$
- Find x_1 and x_2 where

$$x_1 = x_l + d$$

$$x_2 = x_u - d$$

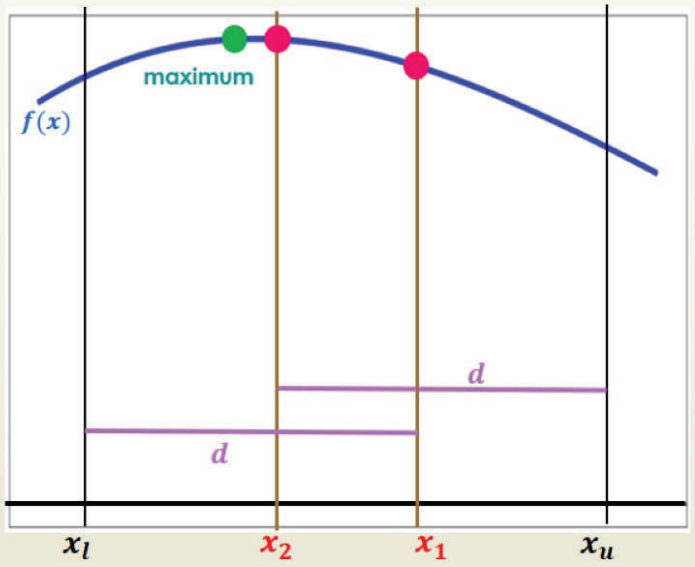
$$d = (\phi - 1)(x_u - x_l)$$

$$\phi = \frac{1 + \sqrt{5}}{2} = 1.6180339$$



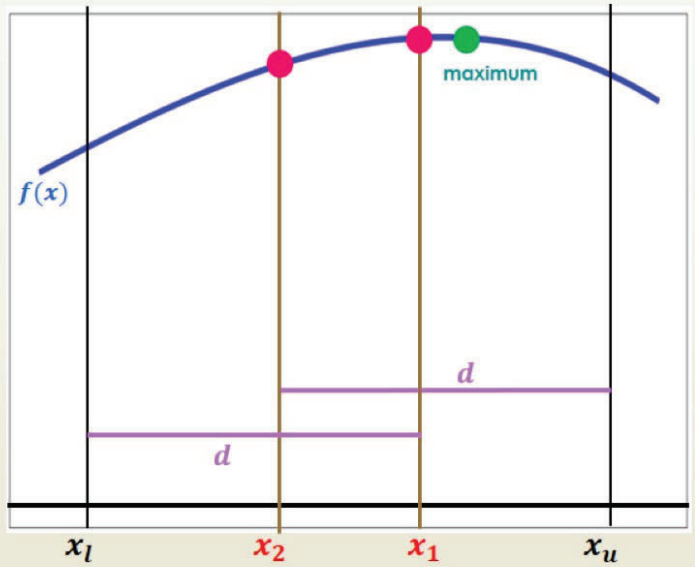
Golden Section Search (maximum)

- If $f(x_2) > f(x_1)$, then
  - the maximum must be in $[x_l, x_1]$
  - set x_2 as the new (estimated) max



Golden Section Search (maximum)

- If $f(x_1) > f(x_2)$, then
  - the maximum must be in $[x_2, x_u]$
  - set x_1 as the new (estimated) max



Golden Section Search

- *This source code is from the textbook, fig7.7*

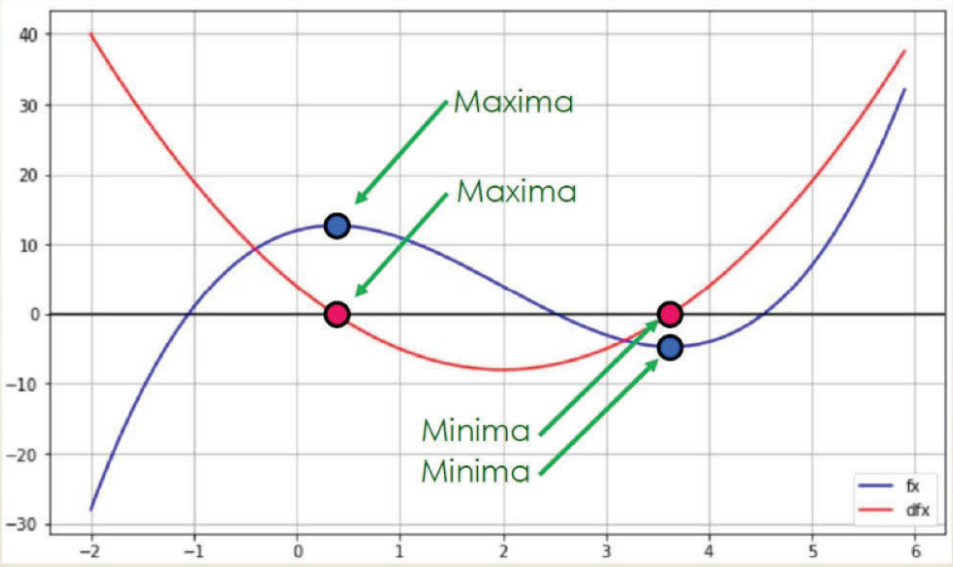
```
import numpy as np

def goldmin(f,xl,xu,Ea=1.e-7,maxit=30):
    """
    use the golden-section search
    to find the minimum of f(x)
    input:
        f = name of the function
        xl = lower initial guess
        xu = upper initial guess
        Ea = absolute relative error criterion
             (default = 1.e-7)
        maxit = maximum number of iterations
                (default = 30)
    output:
        xopt = location of the minimum
        f(xopt) = function value at the minimum
        ea = absolute relative error achieved
        i+1 = number of iterations required
    """
    phi = (1+np.sqrt(5))/2
    d = (phi - 1)*(xu-xl)
    x1 = xl + d ; f1 = f(x1)
    x2 = xu - d ; f2 = f(x2)
    for i in range(maxit):
        xint = xu - xl
        if f1 < f2:
            xopt = x1
            xl = x2
            x2 = x1
            f2 = f1
            x1 = xl + (phi-1)*(xu-xl)
            f1 = f(x1)
        else:
            xopt = x2
            xu = x1
            x1 = x2
            f1 = f2
            x2 = xu - (phi-1)*(xu-xl)
            f2 = f(x2)
        if xopt != 0:
            ea = (2-phi)*abs(xint/xopt)
            if ea <= Ea: break
    return xopt,f(xopt),ea,i+1
```

```
(xopt, yopt, ea, index) = goldmin(lambda x: (x**2)+2*x,-3,5,.05)
```

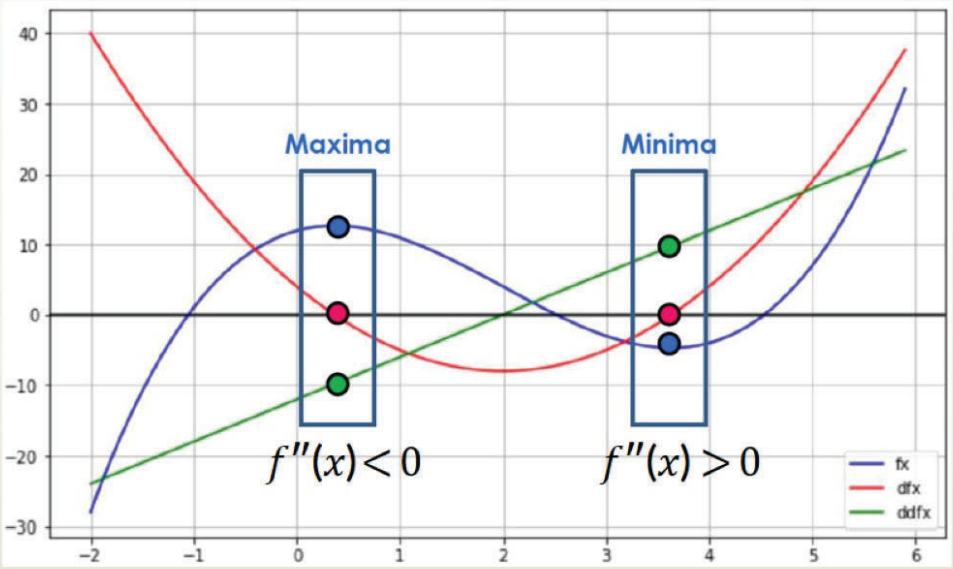

Finding the optimum using root finding methods

- Finding maxima/minima of $f(x)$ is the problem of root finding on $f'(x)$



Finding the optimum using root finding methods

- To determine whether x where $f'(x) = 0$ is min or max, look at the sign of $f''(x)$



Finding the optimum using root finding methods

- Compute the first and second derivatives of the function $f(x)$
- Then, apply root finding methods on the first derivative $f'(x)$
- Evaluate $f''(x)$ at an approximate root to determine if the optimum point found is minima or maxima

```
import sympy as sp
# a Python library for symbolic mathematics

# define a symbolic expression
x = sp.Symbol('x')
fx = (x**2)/9 - 2*sp.sin(x) - 1

# compute its derivatives
dfx = fx.diff(x)
ddfx = dfx.diff(x)

# translates symbolic expressions into func
fx = sp.lambdify(x,fx)
dfx = sp.lambdify(x,dfx)
ddfx = sp.lambdify(x,ddfx)
```